

A new Requirements Engineering approach for Manufacturing based on Petri Nets

Javier Martinez Silva^{*} Raul Javales^{**} José Reinaldo Silva^{***}

^{*} Centro Universitario FEI - Computer Science Department,
Av. Humberto de Alencar Castelo Branco, 3972-B - Assunção, São
Bernardo do Campo - CEP 09850-901
São Paulo/SP, Brasil (e-mail: jmartinez@fei.edu.br).

^{**} Escola Politécnica - Universidade de São Paulo Design Lab. - PMR
- Mechatronic and Mechanical Systems Department - São Paulo,
Brazil (e-mail: rjavales@tnvg.com.br)

^{***} Escola Politécnica - Universidade de São Paulo Design Lab. -
PMR - Mechatronic and Mechanical Systems Department - São Paulo,
Brazil (e-mail: reinaldo@usp.br)

Abstract: Manufacturing systems are going through strategic changes to move from current massive customization production process towards new digital models - also called Industry 4.0. Process planning for this new approaches demand artificial intelligent and requirements modeling that should be formally verified. Therefore, requirements for manufacturing - eventually distributed - processes should be formalized and analyzed to lead to effective smart solutions. Goal-oriented requirements appear as a suitable approach to requirements but still need a formal representation that could deal with discrete distributed plants. Finding such representation is a key issue to model and verify requirements. This paper presents a method and a framework to put together goal-oriented requirements and Petri Nets, as an alternative to the requirements for manufacturing systems capable to deal with digital twins. A classic example associated to a car manufacturing plant is also presented to illustrate the method.

© 2019, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Requirements Engineering, goal-oriented requirements, requirements modeling, manufacturing design, Petri Nets.

1. INTRODUCTION

Manufacturing systems are facing good challenges since the beginning of this century, which have been analyzed following different vectors: the modernization of issues and concepts Kusiak (2017), the relation with human workers Gershwin (2017) or by structuring and architectural approaches Silva and Nof (2018); Dutra and Silva (2016); Silva and Nof (2015). In fact, the real challenge is to design a manufacturing process that are meant to couple with humans not just in the production process but also with the customer. Service-oriented manufacturing is then the target.

In any case, the early processes for manufacturing design, meaning, the requirements engineering, is a key issue to achieve this coupling. Manufacturing Requirements Engineering (MRE) consists also from stages of eliciting, modeling and validation/verification and rely on a formal approach inserted in the very beginning of the process.

This paper proposes a new approach for the design of manufacturing based on the use of model-driven engineering and supported by a framework called ReKPlan (Requirement Engineering in KAOS for Planning) which combine the use of goal-oriented requirement in KAOS and Petri Net. This method can support distributed architectures for manufacturing such as Product-service Architecture

(PSA) Silva and Nof (2018); Dutra and Silva (2016); Silva and Nof (2015).

Section 2 presents GORE Methods and KAOS diagrammatic modeling to requirements. Section 3 will show basic concepts of Petri Nets its approach to analyze requirements. A case study to manufacturing systems is taken from Roade Challenge with a Car Sequencing problem. Finally, some concluding remarks and contributions to further work are presented.

2. GORE METHODOLOGIES FOR MANUFACTURING DESIGN

According to (Cailliau, 2018) efficient requirements engineering (RE) are a key issue to achieve good design systems. A consistent set of requirements that merge different viewpoints and combine humans and hardware resources to provide good product-service can lead to specifications that would guide the design of more flexible solutions, specially if the target is a distributed production arrangement. The relationship between processes, sub-processes and humans resources could provide traceability which is a key issue to maintenance. For all these reasons the proposed approach is driven by objectives instead of functionality.

The key activities to model requirements are: *i*) elicitation; *ii*) modeling and *iii*) formalization/verification. GORE

(Goal-Oriented Requirement Engineering) is an approach developed by Jonhn Miloupolus and formalized by Axel Lamsweerde (Lamsweerde, 2009) which is growing very fast to engineering requirements in software engineer, systems and manufacturing design. Objective-oriented requirements suit better the challenge of production systems that are meant to be anthropocentric and provide a good coupling with the final user to achieve a good value co-creation Silva and Nof (2018, 2015).

KAOS (Keep All Objectives Satisfy) diagrams can be formalized in Linear Tree Logic, which is a good representation to model process and check consistency. However to distributed systems this approach is lead to something close to the product of automaton and is not suitable to large systems. It will describe briefly the main elements of KAOS diagrams to introduce in the following the discussion transferring its semantics to a different formal representation.

2.1 KAOS Graphic Representation

Basically, a KAOS goal diagram is a tree in which all nodes represent goals and the edges represent relations (such as composition, refinement, dependency, restriction, etc.). The root of the tree models the main goal, which is an abstraction of the system - graphically represented by a parallelogram . Fig. 1 shows the basic elements of KAOS diagrams.

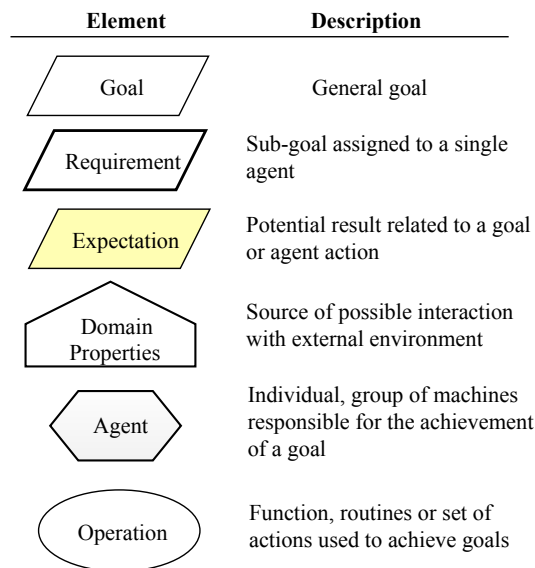


Fig. 1. Main elements of Goal Diagram

Goals should be achieve by *requirements* which are associated to operations. Requirements should be demanded or fit expectations from specific users or worker (a human resource in the process). In any case it is possible to detach different viewpoint in the requirements representation and well as class agents to whom requirements should attend. That is a very important issue since it is the basis for traceability, which is also important to maintenance.

According to Silva et al. (2018), the key difference between the modeling using KAOS diagram or UML is the fact that while the first should integrate all information in four

diagrams the last would require up to thirteen structural diagrams and twelve behavior diagrams even taking in account that much of this is composed by the same information. If time is include - to schedule activities - and the allocation of resources, the functional approach will become a greater challenge Hackel and Taentzer (2018).

Besides, in what concerns formalization, KAOS method and tools direct requirements modeling to Linear Temporal Logic (LTL).

A goal may be described as a valid state, derived from the general behavior of the system. Separately, each sub-goal can emerge from different course of actions but converge to the main goal. Such behaviors can be represented as paths in a graph or by as a combination of different automata. A formal representation prescribed by KAOS method is based in LTL, but it could also be represented by an state-transition formal representation.

A transition could be represented formally in terms of LTL sentences such as:

$$C \Rightarrow \Theta T$$

where C is a current condition, T is a target condition and Θ is one of the LTL operators represented in Table1.

Table 1. Temporal Logic Operators

Operator	Description
\bigcirc	In the next state
\diamond	Eventually in the future
\square	Always in the future
$\bigcup_d \leq$	Hold until d is true

3. USING PETRI NETS FORMALISM TO ANALYZE REQUIREMENTS

A transition system is characterized by the change of state due to the occurrence of an action - or other stimulus. Modeling this evolution is usually done using a directed graph in which the nodes represent states and the edges transitions. Thus, it is possible to define semantic models that describe systems behavior usign graph oriented formal representations. For this, a set of state variables is defined which allows the construction of different abstractions. Transition always specify system evolution state by state, up to a desired final state. This behavior could be reproduced by a digital twin. Any deviation from this behavior could generate an alarm or the start a detailed verification process.

Therefore, the design of such system would demand the use of a formal representation to model requirements, to provide requirements verification, to design specifications and behavior verification. Petri Net formalism has the advantage to fit all these stages. There are several references about about Petri Nets, from introductory texts Murata (1989) up to modern revisions Zhou et al. (2018).

In the scope of Petri nets, the focus of this proposal is High Level Petri nets, GHENeSys approach specifically. GHENeSys stands for General Hierarchical Enhanced Net System, and is an unified petri net defined in 2009(Silva and del Foyo, 2012) aiming to use this formalism in the design process of automated systems.

Figure 2 illustrates the main graphic elements of GHENeSys net including extensions:








Element	Name
	Box
	Activity
	Pseudo-Box
	Arc
	Enabled Arc
	Macro-Box
	Macro-Activity

Fig. 2. GHENeSys is a unified net that incorporate - formally - extensions to Place/Transition and High-Level Net definitions as Pseudo-box, to represent observable but not controllable events and hierarchical elements (macro-box and macro-activities).

The proposal is to transfer requirements represented in goal-oriented diagrams (KAOS) to Petri Nets to support requirements verification and traceability with further solution behavior also represented in Petri Nets. Transference to Petri Net can be done by direct matching the diagram structures or using a KAOS Markup Language (KML) to represent KAOS diagrams and matching that to PNML (Petri Net Markup Language) Hillah et al. (2010). In this paper we will use a markup language associated to the unified extended Net GHENeSys (General Hierarchical Enhanced Net System) depicted in Fig.6.

4. ROADEF: CAR SEQUENCING PROBLEM

The *Car Sequencing Problem*¹ challenge was propose by Nguyen (2005). The main goal is to control the daily production of a car manufacturing.

Indeed, this challenge is a realistic manufacturing planning problem that encompasses interesting features such as planning with resources, sequencing, optimization and flexibility making this problem a good example for the proposed approach.

For instance, the Assembling process must be flexible enough to fulfill special and personalized demands in requirement inserted by co-design processes Dutra and Silva (2016), introduced by specific users: sunroof, special wheel types, air conditioning, etc. Customer orders are sent to car factories in real-time and the factory must assign the production to deliver cars in the proper deadlines, and according to constraints and production line capabilities. A car sequence must be established daily and its accuracy has a direct effect on the amount and quality of manufactured cars. A more detailed description of this challenge can be found in Nguyen (2005).

A digital system that control the Assembly and the Sequence lines and put the together collaboratively Silva

and Nof (2018) must have its requirements formalized and validated, including the information about requirement responsibility. As mentioned before, KAOS could be used to model requirements, but it is also necessary a formal approach based on a transition system to capture the dynamic, suitable perform a formal verification, based o graph properties. Thus, it is crucial to transfer KAOS diagrams to Petri Nets.

5. REKPLAN: TRANSFERRING KAOS DIAGRAMS TO PETRI NETS

ReKPlan was conceived as a complementary system to act together with any environment that generate a KAOS diagrams. First of all, a markup language was defined to represent KAOS diagrams called KML (KAOS Markup Language). Generally, process of semantic transference from the diagrams to Petri Nets can be summarized as a matching between KML and PNML (Petri Nets Markup Language) Hillah et al. (2010). If the extended environment GHENeSys is used the matching will be between KML and GNML (GHENeSys Markup Language) Silva and Silva (2017). Fig. 3 shows part of the modeling for RoadeF, specifically to the car sequencing to fit direct orders from the customers. Following, Fig. 4 shows an example that request the assembling of a car which body was just painted were the goal was directly translated to LTL (Linear Temporal Logic) also associated with an object diagram.

The object-oriented approach is derived from a main goal - shown in Fig. 3 - which is the direct reaction to a car sequence request: “A request is ready for delivery when a order was received”. From this goal and applying successive refinements using milestone-driven technique (Darimont and Van Lamsweerde, 1996) (Van Lamsweerde, 2004) new sub-goals are generated describing all stages of the manufacturing process - reduced here to body setup, painting and assembling.

Refinement ends when sub-goals are linked directly to system agents defined in the associated object diagram. Here the modeled agents are: *Transporter*, related to car transportation from one area to another; *Assembler*, responsible for assembling car units; the *Painter* related to the painting process; and finally *Operator* who sorts and groups cars at each stage (sequencing) to fit paint capacity, associated to the number of cars that can be painted and to the availability of paint on the painting gun - respecting the special features that some cars must satisfy.

Figure 4 shows a sub-goal generated from the primary objective through successive refinements and its proper formalization in LTL. Anticipating formalization can also help to check individual goals while eliciting requirements before refining them. Objects - declared by the object diagram can stand for human resources, hardware or raw material. Objects referenced by more than one goal are not re-declared in the Objects diagram, as for instance, the goal “Cars painted when a order was received”, implies the use of a *SprayGun sg*, to specify a *Color color*, to use an available painting area *Painting Area pa*, and to use a *Painter painter*. All those resources must be taken into account to provide a service to the object *Car, c*, and

¹ All information about the challenges in ROADEF can be found in challenge.roadeF.org. All problems were sent by real companies without some details and information they think to be classified.

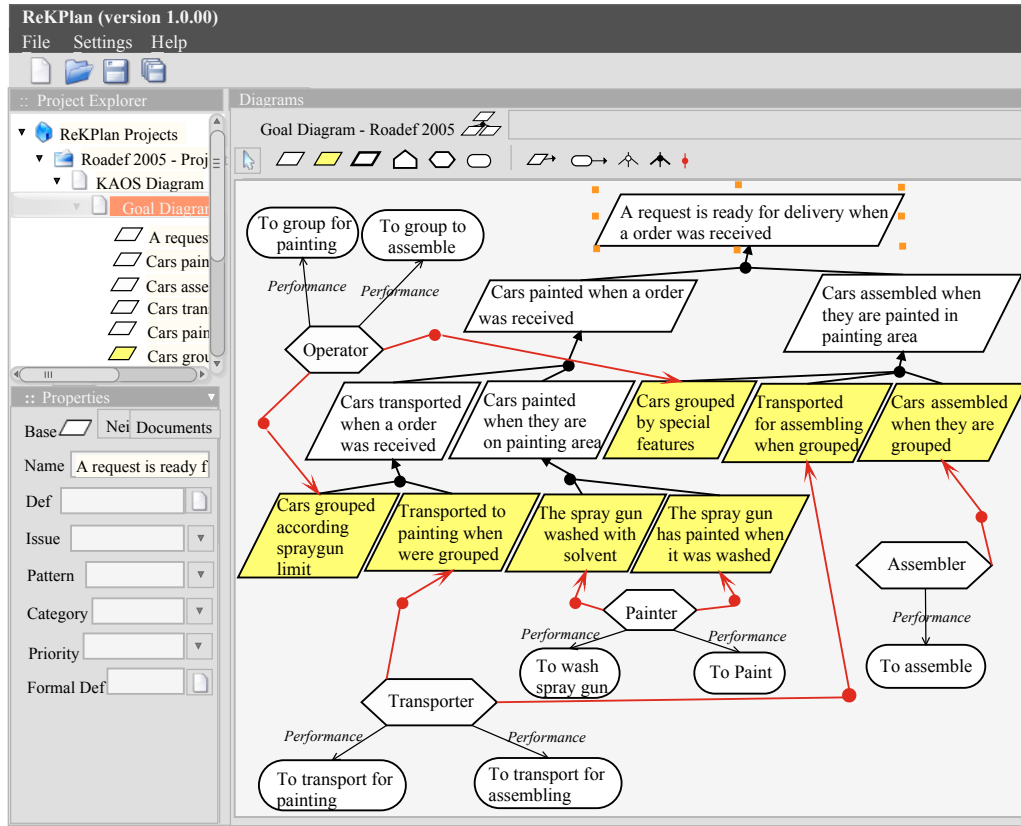


Fig. 3. Goal Model for the Car Sequencing Problem using RekPlan tool

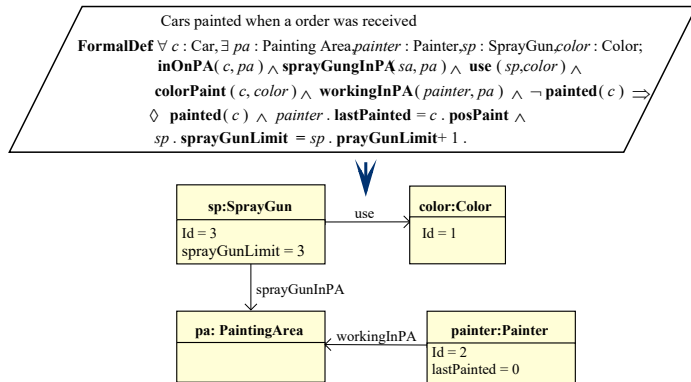


Fig. 4. Objects generated from Linear Temporal Logic sentences associated of "Cars painted when a order was received" goal

achieve the goal "Cars assembled when they are painted in painting area".

All these objects are modeled in ReKPlan as long as they are needed to be associated with goals during the requirements modeling of manufacturing behavior. The resulting object diagram is shown in Figure 5.

The summary of the KAOS goals for the *Roadef: Car Sequencing Problem* with its formalization using LTL can be found in Table 2.

Semantic transference from KAOS diagrams to Petri Nets can be done by representing the diagrams in KML and using GNML to transfer these diagrams to Petri Nets in a unified net. For the goal diagram shown in Fig. 3 the corresponding Petri Net is in Fig. 7. Special emphasis has the extension elements such as the enabling gates (filled in gray) and the hierarchical elements (a circle with a rectangle inside). Hierarchical elements enhance the possibility of reusability and also reinforce modularity, and can be explored when adaptations are necessary and concentrated in this modules.

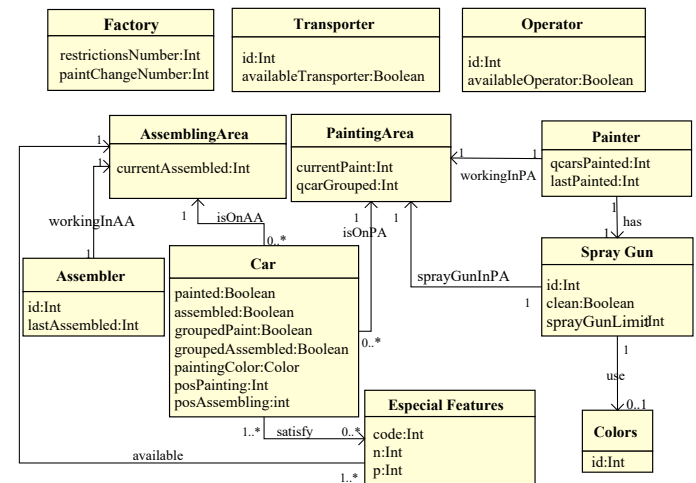


Fig. 5. Objects diagram for Car Sequencing Problem

Figure 6 shows some of the main rules for this transference (Silva and Silva, 2015). Here, Figure 6a) shows the translation for some basic elements of goal model, while Figure 6b) shows the translation for AND/OR refinement to structural net components, and the structural component for milestone-drive refinement (Darimont and Van Lamsweerde, 1996; Van Lamsweerde, 2004). Once requirements model is transferred to Petri Nets (or to an extended version) it is possible to analyze the net model looking for desired properties. The primary check is using reachability trees or discrete simulation to check behavior prospective system. Requirements meant to specify what the problem to be solved, or, concerning manufacturing systems, which behavior is intended.

Other properties to be analyzed is safeness and deadlock freedom. There are algorithms to do these analyses and the result of that using the GHENeSys environment is shown in Fig.8. There is still the possibility to include invariant analysis. That add few information in this specific example of Car Sequencing and were not included in this work.

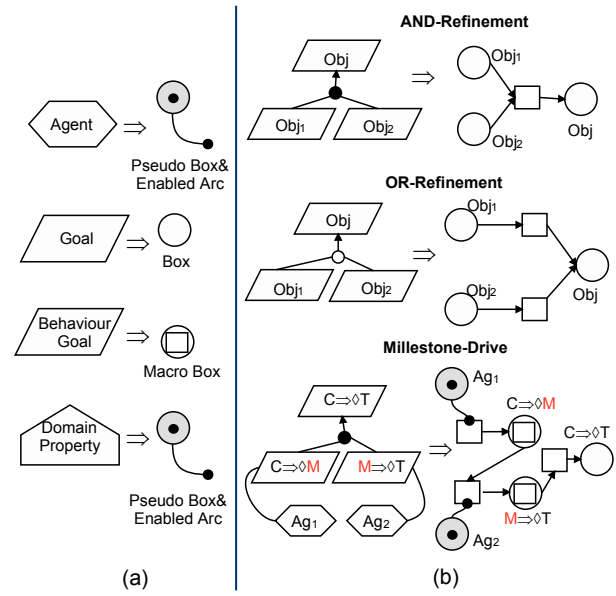


Fig. 6. a) Semantic Translation: basic elements. b) Semantic translation for refinements.

Table 2. LTL sentences associated to some goals of Goal Diagram

Goal	LTL Sentences
Cars painted when a order was received.	$\forall c:Car, \exists pa:PaintingArea, painter:Painter, sg: SprayGun, color:Color; isOnPA(c,pa) \wedge sprayGunInPA(sg,pa) \wedge use(sg,color) \wedge paintColor(c,color) \wedge workingInPA(painter,pa) \wedge \neg painted(c) \wedge c.posPainting = painter.lastPainted + 1 \Rightarrow \Diamond painter.lastPainted = c.posPainting \wedge gs.sprayGunLimit = gs.sprayGunLimit + 1 \wedge painted(c).$
Cars assembled when they are painted in painting area.	$\forall c:Car, \exists ass:Assembler, aa:AssemblingArea; painted(c) \wedge isOnAA(c,aa) \wedge groupedAssembled(c) \wedge workingAA(ass,aa) \wedge \neg assembled(c) \wedge c.posAssembling = ass.lastAssembled + 1 \Rightarrow \Diamond mnt.lastAssembled = c.posAssembling \wedge assembled(c).$
Cars grouped according spray gun limit.	$\forall c:Car, \exists op:Operator; \neg painted(c) \wedge \neg assembled(c) \wedge availableOperator(op) \wedge c.posPainting = 0 \Rightarrow \Diamond groupedPaint(c).$
Cars grouped by special features.	$\forall c:Car, \exists op:Operator; painted(c) \wedge availableOperator(op) \wedge \neg groupedAssembled(c) \Rightarrow \Diamond groupedAssembled(c).$
Transported to painting when were grouped.	$\forall c:Car, \exists tra:Transporter, \exists pa:PaintingArea, sg: SprayGun; groupedPaint(c) \wedge \neg painted(c) \wedge \neg isOnPA(c,pa) \wedge pa.currentPaint < sg.sprayGunLimit \Rightarrow \Diamond isOnPA(c,ap) \wedge pa.currentPaint = pa.currentPaint + 1 \wedge c.posPainting = pa.currentPaint.$
Transported for assembling when grouped.	$\forall c:Car, \exists tra:Transporter, aa:AssemblingArea; \neg assembled(c) \wedge \neg isOnAA(c,aa) \wedge availableTransporter(tra) \wedge groupedAssembled(c) \Rightarrow \Diamond isOnAA(c,aa) \wedge aa.currentAssembled = aa.currentAssembled + 1 \wedge c.posAssembling = aa.currentAssembled.$
The spray gun washed with solvent.	$\forall sg: SprayGun, \exists painter:Painter; \neg clean(sg) \wedge has(painter, sg) \wedge painter.qcarsPainted > 0 \Rightarrow \Diamond clean(sg) \wedge painter.qcarsPainted = 0.$
The spray gun has painted when it was washed.	$\forall c:Car, sg: SprayGun, \exists painter:Painter; has(painter, sg) \wedge clean(sg) \Rightarrow \Diamond \neg clean(sg).$

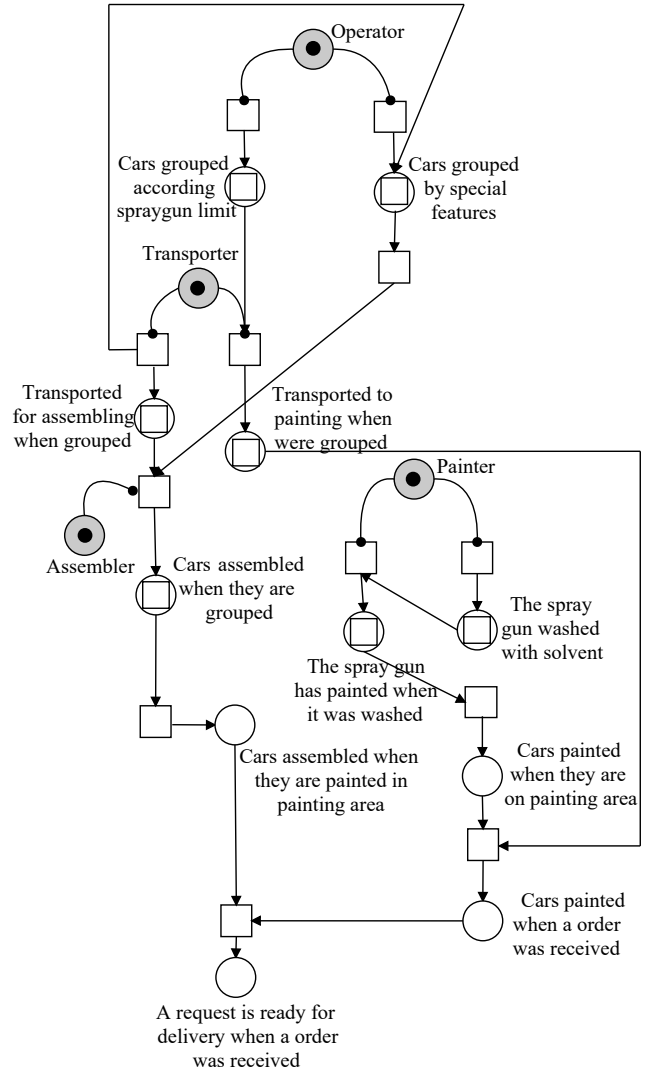


Fig. 7. Petri Net of Car Sequencing Problem.

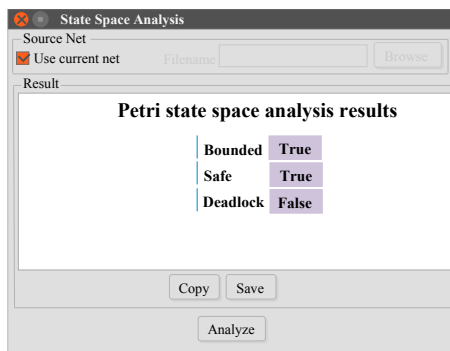


Fig. 8. Petri Net analysis for the Car Sequencing Problem.

6. CONCLUSION

There is enough evidence in the literature that manufacturing is migrating from a model based on big production sites concentrated in a specific place, demanding a logistic that make human and other general resources to converge to the same local, and requirement specif and huge amount of energy supply, to a distribute and much more flexible model. This process is not going fast and several adaptations should anticipate big changes.

However, those intermediary changes will also demand more sophisticated design, which also demand approaches that goes beyond workflow modeling. In fact, the new approach is service-oriented and thus depend on a systemic approach that begins with requirements engineering. A method was proposed to explore an alternative approach to manufacturing based goal-oriented requirements and in a formal model based on Petri Nets, which is also used to model workflow putting together old and new approaches. It is a model that could be adapted to the transition phase and also be used to model full service-oriented digital manufacturing (Martinez and Silva, 2015).

It starts with modeling realistic problems, one of which is presented in this article and are now applying the method to larger problems. Besides that, methods from Artificial Intelligence planning and scheduling are being inserted to the method to address time-slice dependent process and real-time applications.

REFERENCES

- Cailliau, A. (2018). *Software Requirements Engineering: A Risk-Driven Approach*. Ph.D. thesis, UCL-Université Catholique de Louvain.
- Darimont, R. and Van Lamsweerde, A. (1996). Formal refinement patterns for goal-driven requirements elaboration. In *ACM SIGSOFT Software Engineering Notes*, 179–190. ACM.
- Dutra, D.d.S. and Silva, J.R. (2016). Product-service architecture (psa): toward a service engineering perspective in industry 4.0. *IFAC-PapersOnLine*, 49(31), 91–96.
- Gershwin, S.B. (2017). The future of manufacturing systems. *International Journal of Production Research*, 56(1-2), 224–237.
- Hackel, R. and Taentzer, G. (eds.) (2018). *Graph Transformation, Specifications, and Nets*, volume 10800 of *Lecture Notes in Computer Science*. Springer.
- Hillah, L., Kordon, F., Petrucci, L., and Tréves, N. (2010). Pnml framework: An extendable reference implementation of the petri net markup language. In *Applications and Theory of Petri Nets*, 318–327. Springer.
- Kusiak, A. (2017). Smart manufacturing. *International Journal of Production Research*, 56(1-2), 508–517.
- Lamsweerde, A.v. (2009). *Requirements Engineering: From System Goals to UML Models to Software Specifications*, volume I. Wiley. doi:2008036187.
- Martinez, J. and Silva, J.R. (2015). Combining KAOS and GHENeSys in the requirement and analysis of service manufacturing. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 48(3), 1634–1639. doi: 10.1016/j.ifacol.2015.06.320.
- Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4), 541–580.
- Nguyen, A. (2005). Challenge roaddef 2005: Car sequencing problem. *Online reference at http://challenge.roadef.org/2005/files/suite_industrielle_2005.pdf, last visited on August of 2016*, 23.
- Silva, J.R. and del Foyo, P. (2012). *Timed Petri Nets*, chapter 16, 359–372. *Petri Nets: Manufacturing and Computer Science*. Intech.
- Silva, J.M. and Silva, J.R. (2015). Combining kaos and ghenesys in the requirement and analysis of service manufacturing. *IFAC-PapersOnLine*, 48(3), 1634–1639.
- Silva, J.M. and Silva, J.R. (2017). Gore methods to model real world problem domains in automated planning. In *Proceeding of the Brazilian Symposium on Automation*.
- Silva, J.M., Silva, J.R., Salmon, A.Z.O., and del Foyo, P.M.G. (2018). Requirements engineering at a glance: Comparing gore and uml methods in the design of automated systems. In *Congresso Brasileiro de Autômatia CBA2018*.
- Silva, J.R. and Nof, S.Y. (2015). Manufacturing service: from e-work and service-oriented approach towards a product-service architecture. *IFAC-PapersOnLine*, 48(3), 1628–1633.
- Silva, J.R. and Nof, S.Y. (2018). Perspectives on manufacturing automation under the digital and cyber convergence. *Polytechnica*, 1(1-2), 36–47.
- Van Lamsweerde, A. (2004). Elaborating security requirements by construction of intentional anti-models. In *Proceedings of the 26th International Conference on Software Engineering*, 148–157. IEEE Computer Society.
- Zhou, L., Zhang, L., Zhao, C., Laili, Y., and Chu, L. (2018). Diverse task scheduling for individualized requirements in cloud manufacturing. *Enterprise Information Systems*, 12(3), 300–318.